STRATEGY RESEARCH PROJECT

A HALTER AND A LEAD ROPE
A WARNING TO NEW GOVERNMENT SOFTWARE DEVELOPMENT MANAGERS OF EXPERIENCES OF THOSE WHO HAVE GONE BEFORE THEM

BY

LIEUTENANT COLONEL ROY J. JENSEN

Senior Service College Fellow, United States Army
THE UNIVERSITY OF TEXAS

USAWC CLASS OF 1999

U.S. ARMY WAR COLLEGE, CARLISLE BARRACKS, PA 17013-5050

UNCLASSIFIED

USAWC Senior Service College Fellowship Research Project

A Halter and A Lead Rope

A Warning to New Government Software Development Managers of Experiences of Those Who Have Gone Before Them

by

LTC Roy J. Jensen

Walter B. LaBerge, Ph.D.
Project Advisor

US Army War College
Senior Service College Fellow
University of Texas
1998-1999

Institute for Advanced Technology
UNIVERSITY of TEXAS at AUSTIN

UNCLASSIFIED

Table of Contents

Forward

There is a saying among old ranchers and horse trainers that, if you start horse trading with someone who knows more about horses than you do, eventually all you end up with is a halter and a lead rope. That sage advice is more than a tribute to my father (who happened to be an old rancher and horse trainer), but it also describes the general state of affairs with the government and their relationship to the software development organizations.

After dedicating a portion of my misspent youth to digging fence post holes in the hundred degree plus California sun, it became intuitively obvious to me that working in an air conditioned computer machine room was better (not the toughest decision I've ever made). Yet the lessons I observed in my youth transcend both environments. Whether dealing in horses or having computer software developed, the rules of horse trading (contracting) have remained basically the same since the beginning of humanity – the person who has the most knowledge of the subject will generally get the best deal. When both parties are on equal footing the chances of a fair and equitable contract are increased. That contract becomes the makings of a partnership. The partnership generates a mutual trust that allows both parties to push in the same direction which is especially necessary for the success of software projects. Moreover, I believe that education of the government and military software development managers should match a level as closely as possible to their civilian counterparts as the only means to this end.

Software development is difficult, complex and *there are many layers to its onion.* Software development is a business that is typified by projects that must fail before

they succeed and their management reflects the trial and error nature of the programming craft. As the millenium comes to a close and the complexity of software has continued to increased to a point where Ockham's Razor (when faced with multiple solutions to a problem – the simplest one is usually the best) is no longer the rule, but actually its antithesis. Where software development project failures are the norm, there is an environment that allows good people with good intentions to have their career's destroyed. In this culture, I felt it necessary to provide a warning to new government managers who embark on these projects. I have attempted to provide a rudimentary set of tools and techniques for the software development process. These tools and techniques that are explained in detail in the body of the paper include:

- The application of commercial software development techniques, specifically "Best Practices" to military projects.
- The use of contractor incentives to implement "Best Practices" on military projects.
- The use of the Software Program Manager's Network as a resource to assist in the implementation of "Best Practices".
- The isolation software rework.
- Insuring that Department of Defense information and acquisition professionals are trained and educated in software development management techniques.

Finally, with apologies to the Surgeon General of the United States, **WARNING**: Software Development Projects Can Be Hazardous To Your Career.

I. Introduction

Joint Vision 2010 stresses "**information superiority** – the capability to collect, process and disseminate . . . information . . ."[i] Future military operational successes, if not the entire strategic direction of the country, will depend on reliable software systems to achieve this goal. The main contribution of this study is the premise that, by evaluating and implementing the successful commercial software development processes and techniques and educating its managers, the Department of Defense should be able to amass a healthier, more reliable product while realizing cost savings and scheduling adherence.

Department of Defense systems, regardless of purpose, are becoming increasingly dependent upon software as a necessary component of their operation. Software size and complexity has grown as the intricacy of the systems it supports has increased. Norm Brown, founder and executive director of the Software Program Manager's Network, has stated that: "Software has become a major cost, schedule, and performance driver of virtually all DoD weapons, command and control, and information systems . . . it has become alarmingly apparent to DoD executives and the U.S. Congress that the software tail wags the system dog."[ii] There is every indication that this dependence on software will continue to grow as the complexity of operations and equipment, as well as the use of simulations and modeling of defense systems, continues to evolve. Proper management of software acquisition thereby becomes one of the critical points in scheduling, costing and maintaining any defense system.

The simulations and modeling community is particularly prone to the inherent problems of software development process. Simulations and modeling programs, by

their very nature, are complex because they must embody all possible permutations of the situation they reflect. Additionally, they are almost entirely dependent upon the software that drives them.

This paper will scrutinize the present state of both commercial and military software development where failure of new projects and programming for poor performance is the norm. In addition to scrutinizing the current state of software development, this study will investigate commercial experiences, the "Best Practice's"[iii] methodology and their applicability to military projects. The fundamental approach to this investigation will be to analyze current management of commercial and Department of Defense software acquisition programs. This paper will encompass the following three main subject areas:

(1) Past performance failures;

(2) "Best Practices", examples and techniques;

(3) General recommendations on how the Army can improve the software development environment.

II. Background

Computers are machines that are capable of three things: accepting structured input, processing the input according to a set of prescribed rules, and producing the results as output of some fashion.[iv] Computers are comprised of two basic components: hardware, the physical components; and software, the instructions or programs that tell the hardware what to do.

The evolution of digital hardware design, formally called computer hardware engineering, is a highly advanced, stable and mature technological discipline. Since the product of the process only has to accept either a zero or a one as input, it is simple in

comparison to the complexities of software. Thereby, it lends itself to the more established engineering methodologies with their inherent management, control and predictive features. In fact, "Moore's Law," named after Gordon Moore, cofounder of Intel Corporation, predicts that microprocessors will double in size (transistor density) and processor performance (speed) every 12 to 18 months. For the past 30 years "Moore's Law" has held true. This equates to approximately a two order-of-magnitude increase every 10 years or 1,000,000-fold increase since the introduction of the microprocessor (the 4004 chip) to the current Pentium III chip.

> *If the automobile industry advanced as rapidly as the semiconductor industry, a Rolls Royce would get 500,000 miles per gallon of gas and would be cheaper to throw away than to park it.*
> *Gordon Moore, President Intel Corporation*[v]

Software engineering (as a defined discipline), on the other hand, is relatively new and tends to play a game of catch up with the hardware component. To illustrate this point, in 1997 the Institute of Electrical and Electronic Engineers (IEEE) finally defined software engineering: "the application of systematic, disciplined, quantifiable approach to the development, operation and maintenance of software."[vi] With the hardware component doubling capacity and performance every 12 to 18 months, it's understandable to see why, even with a conservatively estimated two-year development cycle, that software is always behind its hardware component.

To further compound the problem of software maturation lagging behind the hardware component, software development is at the mercy of the consistent drift or change in programming languages, as well as the variation in computer operating systems. Within the last 50 years, the higher level languages that are used to interface with and load instructions into the computer have gone through multiple iterations from

Fortran to Cobol to Ada to C to C++ with many variations in between. Operating systems are based on the computer size and currently vary from MVS for mainframes through UNIX and AIX for midranges to DOS, Windows and NT for microcomputers, to name only a few. Each one of these languages and operating systems have their own features that are designed to take advantage of the latest hardware innovations and the preferences of the current generation of programmers.

However, there is little or no standardization in the syntax between computer languages; this is especially true for operating systems. The obvious problem with so many different languages and operating systems is educating the personnel in their efficient use. The next issue (and possibly the more pressing for the Army) is the requirement to train management to oversee the programming personnel, as well as estimate the project size, cost and scheduling on this fluid platform. This once again puts the government at the back end of the learning curve.

Software, when viewed as a completed engineering product, contains an inherent flexibility or complexity that is not present in its hardware component. Therefore, because of these factors software does not directly lend itself to the more established engineering methodologies with their inherit management, control and predictive features. In fact, Frederick Brooks, who originally coined the phrase, "No Silver Bullet"[vii], has asserted that: " . . . [Because] the complexity of software is an essential property, it does not lend itself to simplification techniques found in other [engineering] disciplines."[viii]

Additionally, software is hard to build, is essentially handcrafted and by its very nature prone to human error. According to Capers Jones, "The problem is that software

has the highest manual labor content of almost any manufactured item in the second half of the 20th Century."[ix] David Parnas, a leading software engineer on the Strategic Defense Initiative ("Star Wars") project, has carried the handcrafted concept even further by stating that: "Software development is a trial and error craft. People write code without any expectation that they will be right the first time".[x] Finally, Wayt Gibbs in his 1994 *Scientific American* article "Software's Chronic Crisis" stated: "Software is still handcrafted by artisans using techniques they can neither measure nor consistently repeat."[xi] In fact, the concept of software being "built" comes into question – software can be designed, but it can not be built in the physical sense. However, the use of the phrase to "build software" still remains in the common engineering lexicon to describe software development.

> *Several factors contributed to the [software development] situation. First, the invisible nature of both the work process and its product made software projects very difficult to manage and predict. Second, the explosive growth of the use of computers created demand for new programmers, most of whom were self-taught on the job; and frequently, low productivity and poor quality resulted. Third, there was little idea then how to train programmers properly. Fourth, a tradition grew that programmers were secretive craftspersons, whose products, during development, were their own property.*[xii]

The question now becomes who writes software? For those people who have been assigned to the Washington D.C. area (or any major metropolitan area for that matter), have occasional insomnia and a propensity to watch old movies on television, the answer to that question becomes obvious. Every other commercial during the movie is a local "school" advertising to make you a "computer programmer" in just a few short weeks. What is driving this phenomenon?

III. Growth and Education

According to the National Software Alliance, throughout the 1990s the demand for software workers has grown at an unprecedented rate.[xiii] By 1996, virtually every industry in the United States employed software workers – about 1.5 million in all.[xiv] These industries are competing within finite labor constraints, driving up salaries, bonuses, and stock options. Again, according to the National Software Alliance, this is an ineffectual attempt to attract and retain employees and, as the demand for software increases, the demand for qualified software workers will escalate at an even faster rate.[xv] With this much of a demand, it becomes a true sellers market and the quality and level of the training of the workers becomes suspect.

College students graduating with degrees in computer science and software engineering in the United States is declining[xvi]. Besides the obvious problems associated with this decline, there are other problems associated with the graduates. In a recent conversation with Dr. K. Suzanne Barber, the Director of The Laboratory for Intelligent Processes and Systems and Associate Professor of Electrical and Computer Engineering at the University of Texas, the problem became clear – basic software management development techniques were not being taught or utilized in their laboratories. Why? The answer is relatively simple. The university system is educating people to do research – not training managers to oversee large complex software development projects. These projects require managers who have a software engineering background or software engineers with a program management background. (The appendix to this paper describes the National Defense University,

Information Resource College's graduate level Chief Information Officer Certification Program that is attempting to solve this educational paradox for government officials.)

To further complicate this education problem, there is the question of where the top 10% of software development personnel go to work. The answer is **not** the government. The National Software Alliance defines three software worker employment tiers.[xvii] Tier one is composed of software start-ups, boutique software firms and corporate/university research and development organizations; tier two is made up of value-added resellers, consulting firms, software-intensive industries such as IBM and AT&T, aerospace, imbedded software (GM, Boeing), corporate information systems, and applications development. Tier three is dominated by the Department of Defense, federal, state and local governments. Based on the general personality traits (such as independent thinking and a distaste for structure) of "crafts persons" and the educational training described above, it's no small wonder that software workers tend to be drawn to tier one as the most desirable and see tier three as the least desirable.[xviii] Furthermore, regardless of the tier that the individuals are drawn to, the problem of their education in the "engineering" disciplines still comes into question.

> *Creatives are intense. They're always thinking about work. For them there is no such thing as "Miller Time." They are happy to come to work everyday and solve puzzles. . . In general they work for three things. First, the "fun" of creation itself. Second, "admiration" – especially from their peers. Third, the excitement and "glory of taking part in a successful creation.*[xix]

David Parnas' answer to the same question of who writes software is: "*everyone . . . all kinds of engineers . . . commerce/business majors . . . kindergarten teachers, lawn cutters, ...*" [xx] He additionally states that: "When discussing the risks of using

computers, we rarely mention the most basic problem: most programmers are not well educated for the work they do."[xxi] "Detailed knowledge of arcane system interfaces and languages is no substitute for knowing how to apply fundamental design principles."[xxii] In his lecture, "Software Engineering: An Unconsummated Marriage", David Parnas uses the well-publicized year-2000 (Y2K) problem to illustrate his point.[xxiii]

The basic crux of the Y2K problem is that dates have always been stored internally in the computer as six characters with the "19" being omitted from the year to save storage space. The year 2000 becomes "00" [as does] the year 1900. Since the late 1960s (or 60s if you prefer) programmers have known how to design computer programs so that it is easy to change the amount of storage used for dates. "Nonetheless, thousands of programmers wrote millions of lines of code that violated well-accepted design principles."[xxiv] David Parnas' conclusion: "The simplest explanation: those who designed and approved that software were incompetent!"[xxv] While at first glance this appraisal appears rather harsh, it does portray the software-engineering dilemma. However, another explanation of the Y2K problem could stem from that same independent mentality that is prevalent throughout the programming craft. More importantly, it demonstrates the requirement for software engineering education that merges design, integration, and engineering principles and the incorporation of these trained professionals throughout the project management process.

> *Software is a major source of problems for those who own and use it. The problems are exactly those to be expected when products are built by people who are educated in other professions and believe that*

> *building things is not their "real job" and who are not prepared for this job by a professional education.*[xxvi]

"The essence of software is that it achieves the solution of a complex problem by compounding its complexity (i.e., the algorithms defining the solution are exponentially more complicated than the real-world problems they solve)."[xxvii] This complexity is the main contributing factor to the difficulty in estimating the scope of a software project. As David Parnas stated in 1985: "software is hard to build because it is inherently and necessarily complex". [xxviii]

> *Software entities are more complex . . . than perhaps any other human construct . . . Software systems have orders-of-magnitude more states than computers do . . .* [xxix]

IV. Commercial Software Development

Commercially, more than $250 billion is spent annually on software application development projects. [xxx] The average costs range from of $400,000 per project for small companies (companies with annual revenues of $100 to $200 million) to $2 million for large companies (companies with annual revenues greater than $500 million). [xxxi] Recent reports from the Standish Group International Inc., (a market research and advisory firm specializing in mission-critical software and electronic commerce[xxxii]) profess that only 16.2% of all commercial software development projects that are initiated are delivered on time, on budget and with all the features and functions as initially specified in the requirements statement.[xxxiii] In fact, 52.7% of the projects are considered "challenged": projects that are completed and operational but over budget, over the time estimate, and offer fewer features and functions than originally specified.[xxxiv] The remaining 31.1% of the software projects were cancelled sometime during their development cycle.[xxxv]

Of the software projects that are completed by the largest of the United States organizations, only 42% have the originally proposed functions.[xxxvi] Moreover, 52.7% of the completed projects for these organizations will cost 189% of the original estimates with only 9% completed on time and on budget.[xxxvii]

The bottom line is that, commercially, there was an $81 billion loss to the United States in 1995 due to cancelled projects, with the average Management Information System (MIS) project being one year late and 100% over budget.[xxxviii] The following chart illustrates the point:

Year	Project	Results
1980	International Telephone & Telegraph [xxxix] *4 Switching Systems*	40,000 Function Points $500 Million Lost **Cancelled**
1987	California Department of Motor Vehicles [xl] *Automated Vehicle/Driver's License System*	15,000 Function Points $30 Million Lost **Cancelled**
1989	State of Washington [xli] *Automated Social Service Caseworker System*	7 years to build Failed to meet user needs $20 Million Lost **Cancelled**
1992	American Airlines [xlii] *Flight Booking System*	$165 Million Lost **Cancelled**
1994	Denver Airport Baggage System [xliii]	$27 Million Cost Overrun $55 Million United Airlines $51 Million City of Denver $35 Million Other Airlines $360 Million Delay Costs $37 Million Lost Income $8 Million in Bond Fees $20 User Fee per passenger
1995	FAA Air Traffic Control Modernization [xliv]	$5.6 Million Overrun 8 year schedule slip 2 of 4 systems **Cancelled** Third system reduced by 48% Restructured Cost $6 Billion **Required Capacity Downgrade**

These figures cover only the costs for initial software development, and that is only the beginning of the problem. In a typical lifecycle management cycle for a software system, there is the following percentage cost breakdown: developing

requirements (3%) and specifications (5%), designing (7%), coding (5%), testing (7%), and integrating (6%) of the software system.[xlv] Additionally, the statistics on the complete life cycle cost distribution of software projects show that the remaining 67% of the costs are contained in the maintenance function.[xlvi] Just on face value alone, who in their right mind would knowingly buy any product that has two-thirds of its value located in its maintenance functions?

The statistical breakdown of the maintenance function percentages is even more revealing.[xlvii]

41.8%	Changes in Requirements
17.4%	Changes in Data Formats
12.4%	Emergency Fixes
9.0%	Routine Debugging
6.2%	Hardware Changes
5.5%	Documentation
4.0%	Efficiency Improvements
3.4%	Other

As these statistics indicate, software development projects have traditionally had difficulty in maintaining their requirement's specifications (41.8%), data formats (17.4%), and emergency fixes (12.4%) throughout the development lifecycle. These areas directly translate into avoidable system defects that generate exorbitantly high rework percentages and costs.

> *The hardest single part of building a software system is deciding precisely what to build. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.*[xlviii]

According to Rome Laboratories, 50% of all software errors are oversights or errors in the requirement's analysis process.[xlix] Additionally, requirements errors are more difficult and expensive to correct if they are allowed to permeate throughout the

development life cycle. Samuel DiNitto, in his article *Rome Laboratory*, has noted that: "It is often fifty times more expensive to correct a defect during systems integration [phase] than during requirements analysis."[l] Barry Boehm, the director of the Center for Software Engineering at the University of California, makes an even more alarming observation: "It costs a hundred times more to find and fix a requirements or architecture defect during integration test (or later) than it does to fix the defect when [it is] made."[li] Regardless of which estimation is correct, these observations continue to reveal the need for the use of "Best Practices" methodologies and the education to use them. To illustrate this point, Norm Brown and the Software Program Manger's Network have shown that a small number of high-leverage proven [best] practices can be put in place quickly to achieve relatively rapid bottom-line improvements to overcome these problem.

V. Software Rework

> *Defects that result in rework are one of the most significant sources of risk in terms of cost, delays, and performance . . .. **Because this high level of rework does not reflect positively on developers, it is not often openly reported,** and can be in the 40 to 50 percent range.*[lii]

Christine Davis, of Texas Instruments, in 1996 stated: " . . . the costs show that 40 to 50 percent of your cost – your development cost – is in reworking and getting defects out. And, if you use formal inspections, you can find 80 percent of the defects as they happen."[liii] Paul Solomon of Northrop Grumman agrees: "Our prior experience [with software rework] is in the 40 to 50 percent range."[liv] Finally, Barry Boehm, in *IEEE Software,* confirmed these statistics: "Reworking defective requirements, design, and code typically consumes 40 to 50 percent of total development costs."[lv] The Software Program Manager's Network has done the mathematics.[lvi] Of the $11.7 billion DoD

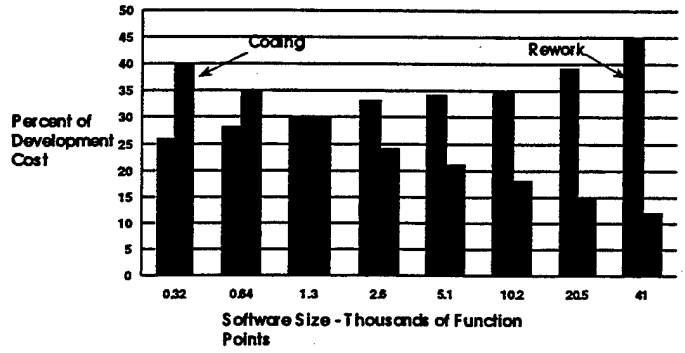
spends for development, $5.1 billion is for rework. Of the $23.7 billion DoD spends for maintenance, $10.4 billion is for rework. Total rework cost is $15.5 billion. To quickly put this figure into perspective, 16.1% savings in rework costs translates to **approximately $2.5 billion or about 480 M1A2 tanks or 2.8 Arleigh Burke destroyers or 27.5 F-16 airplanes.** Finally, a less than 50% savings in the DoD rework costs would buy **all** this equipment.

Current and former senior defense industry executives and program managers made the following statements under a non-attribution policy and that were compiled by the Software Program Manager' Network.[lvii] The comments illustrate the management problems associated with software project management in general, rework in specific:

> *"I'm not worried about this [rework] because the government keeps coming back to fix it."*
> *"Claim rework as new business and log new business bookings for bonuses."*
> *"They give us bonuses for more DoD work – new work, rework, whatever."*

Rework (also known as "scrap rate") problems are the manifestation of the much larger software acquisition problem of project management. Again, what traditional engineering discipline or manufacturing process would tolerate a "scrap rate" in the range of 40 to 50 percent? To repeat the Defense Science Board: "Today's major problems with military software development are not technical problems, but management problems."[lviii] The following chart demonstrates this management problem as a function of complexity of the project:[lix]

Development Costs of Coding & Rework As A Function Of Complexity



* Jones, Capers, *Applied Software Measurement: Assuring Productivity and Quality*, McGraw-Hill, 1991, p142

Does this 40 to 50 percent "scrap rate" continue to be realistic with the current preference for fixed price contracts and Cost As an Independent Variable (CAIV) methodologies within the Department of Defense community? One side of the argument says that there is no printing press to print dollars to cover cost overruns and rework. Programs that suffer from overruns are subject to scaling back or cancellation. Under firm fixed price contracts, software developers should have to absorb their own inefficiencies. The other side of the argument says that, if the government does not or can not correctly estimate the size of the project, then the rework costs can be factored into the contractor's original development costs. Therefore, these costs can become invisible to the Government. Once again, if the person you are horse trading with is better at it than you are, you continue to run the risk of ending up with a halter and a lead rope and education continues to be the answer.

> *The fundamental reason software-intensive developments overrun cost and schedule, with the resulting quality and performance shortfalls, is our [the government's] **inability to estimate** . . . We [the government] also do not account for the amount of **scrap** and **rework** of code involved when a developer has an ad hoc, chaotic development process . . . [the scrap and rework] to be about 44% of every dollar spent* [lx]

VI. Function Point Analysis

Currently, software complexity is measured by the technique *de jour* of Function Point Analysis. This technique is one of the replacements for the antiquated Source Lines of Code (SLOC) method that has been used since the first line of code was written. SLOC merely counts the number of lines of code in the source program. This technique is dependent upon the computer language and the skill of the programmer, both of which are highly variable. With this much variance, SLOC is considered to be an inefficient estimating tool; however, it is still extensively used throughout the government.

Function Point Analysis was initially made public by Allan Albrecht of International Business Machines (IBM) in 1979 as a technique to quantify the functions contained in software in meaningful terms to end users.[lxi] The function point measure is derived in a number of stages using a standardized set of basic criteria. Each function the program performs is given a numeric index according to its type and complexity. The indices are totaled to give an initial measure of size. Incorporating a number of factors relating to the software as a whole, the size is then normalized. The resultant is a singular number called the Function Point index that measures the size and complexity of the software. Function Point Analysis provides an objective, comparative measure to assist in the estimation, evaluation, planning, management and control of software production.

The requirement for a common estimating tool is essential to evaluate the magnitude of a software project. A common tool, such as Function Point Analysis, allows both the government and the contractors to speak the same language and

compare apples to apples, rather than apples to oranges. Additionally, without the ability to independently estimate the scope of the software development project the government is at the mercy of the contractor's estimate. The contractor's estimate may or may not have included such things as rework costs. Without a common estimating tool, this is difficult, if not impossible, for the government to determine. Presently, software development costs in the United States are at about $1000 per function point.[lxii] There is an old saying in among software developers – "It is morally wrong not to separate a naïve end user from his/her money."[lxiii] The need to educate government project management personnel in software estimating techniques and "Best Practices" becomes critical to controlling development cost and scheduling problems.

VII. Software Development

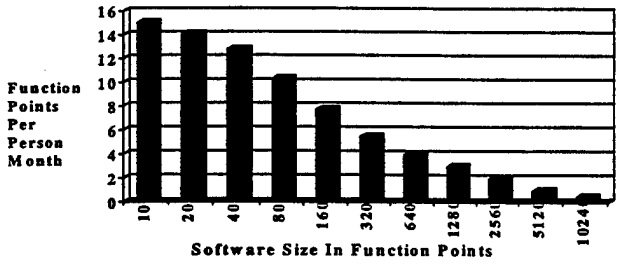
Cetron and Davies in their book, *Probable Tomorrows: How Science and Technology Will Transform Our Lives in the Next Twenty Years*, have stated: "Without software to control it, all this hardware is just scrap metal, plastic, and highly purified sand."[lxiv] Computers without software used to be compared to common mindless household appliances like the toasters and coffee makers of the 1950s. However, with the state-of-the-art appliances currently on the market, even these basic devices have embedded computers that act as control mechanisms that require software programs to control them. Therefore, without software, the computer hardware and the devices they control become inefficient boat anchors.

The ability of software to modify or control the configuration of a hardware device without re-manufacturing generates a versatility that cannot be easily achieved by any other means. Therein lies software's power. But with this power also comes a

complexity that exceeds, according to Frederick Brooks, by orders of magnitude the complexity of the computers and machines they control. [lxv]

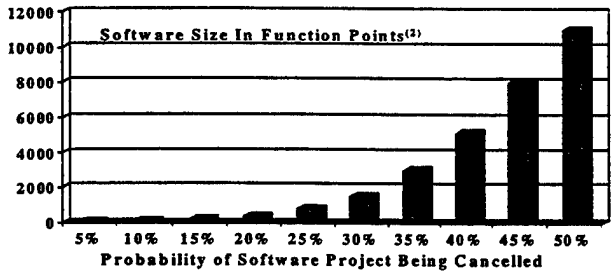
At this point, it becomes a blinding flash of the obvious that the more complex the software system, the more inherent risk associated with the project. The mitigation or sharing of this risk appears to be the main technique to provide incentives to the contractors for the use of the "Best Practices" methodologies. The following graphs [lxvi] provided by the Software Program Manager' Network demonstrate this risk as a function of project size:

PRODUCTIVITY VS. SIZE*



* Capers Jones. Becoming Best In Class. Software Productivity Research. 1995 briefing

% CANCELLED VS.SIZE(1)



(1) Capers Jones. Becoming Best In Class. Software Productivity Research. 1995 briefing
(2) 80 SLOC of Ada to code 1 function point, 128 SLOC of C

Presently, it is purported that the Department of Defense spends approximately thirty-six billion to forty-two billion dollars each year on software research, development and acquisition.[lxvii] Additionally, the Software Program Managers

Network reports that there is an estimated forty-two billion dollars spent annually by the Department of Defense for the development and maintenance of its computer systems with only seven billion dollars of that money going to the purchase of hardware.[lxviii]

However, exact dollar amounts for DoD software acquisitions are difficult, if not impossible, to obtain because of the software acquisition costs that are embedded in the total costs for the projects. In fact, according to the National Software Alliance, the Office of Management and Budget (OMB) does not require the Department of Defense to report what it spends on software that is embedded in weapons systems and in command, control, communications and intelligence (C^3I) systems that are classified as national security systems (NSS). [lxix] Even with that caveat, this estimated amount of forty-two billion dollars still represents 17% of the total Department of Defense dollars. The figure of forty-two billion dollars has been restated by Mr. Tom Crean, President of the Defense Acquisition University, in his keynote address to the Improving Software Acquisition Management Workshop on 13 February 1997.[lxx] Additionally, it is also unrealistic to believe that the dollar values and statistics for large Department of Defense software development projects would vary far from their commercial counterpart's.

Moreover, the Army's battlefield digitization program calls for $2.8 billion in fiscal year 2000 - a $200 million increase over $2.6 billion allocated in fiscal year 1999 for the Force XXI digitization program, according to the February 1, 1999 issue of *Federal Computer Week*.[lxxi] "Force XXI integrates emerging technologies, new doctrine, force organization and quality soldiers to produce a versatile Army capable of dominating

future battlefields."[lxxii] These systems are designed to provide a fully integrated command and control capability from the regional commander-in-chief down to the individual soldier. Commanders at all levels will be able to develop "situational awareness" of friendly and enemy forces and "pierce the fog of war".[lxxiii] The simulation development communities within the Army have addressed these effects and are planning to incorporate them into their software. In fact, they have modified the current meaning of the phase "phenomenology"[lxxiv] to describe them. However, by including these effects in the simulations program we have now added one more facet to the problem, adding to the software's complexity.

> *The need for the U. S. military forces to adapt to new and more diverse military missions is matched by the requirement to meet these challenges within the constraints of available resources. The concurrent explosion in new technologies offers opportunities to innovative assess new ways of addressing these issues . . . Information Age Technologies will provide warfighters with a breath and depth of information unparalleled in military history.*[lxxv]

The obvious conclusion: the success or failure of all future military operations, regardless of scope, becomes unconditionally tied to the success or failure of the supporting technologies. These information age technologies are primarily driven by their software components. Therefore, successful software development and acquisition becomes the linchpin of the military's future. To reiterate, the future military operational successes, if not the entire strategic character of the country, will depend on reliable software systems.

VIII. Best Practices

> *Unfortunately, the military and defense domain has no real incentive for adopting civilian best practices. The DoD itself and the military services are not for profit organizations and if they tend to overspend or develop software in a way that is worse than the civilian sector, so*

long as the fundamental mission requirements are not compromised, there are no overwhelming reason to improve.[lxxvi]

According to a 1994 Defense Science Board task force report, a number of Department of Defense software development projects have repeatedly gotten into trouble or failed because of cost and scheduling problems.[lxxvii] Additionally, they noted that the current Department of Defense software development process is not compatible with current commercial business practices.[lxxviii] In a time of serious cutbacks to the budget, the military may no longer have the ability to tolerate the cost and scheduling overruns that commercial software development processes are addressing. As a consequence, the Defense Science Board recommended that the Department of Defense use the commercial "Best Practices" techniques.[lxxix]

Best Practices. If we don't articulate and distribute winning techniques, we can't repeat our successes. We must capture – in a central electronic place – proven techniques for estimating, staffing, and managing projects, gathering user requirements, controlling prototypes, designing architectures, deploying systems, and more. [lxxx]

What is a "Best Practice"? A "Best Practice" is a management or technical technique that has consistently demonstrated the ability to significantly improve the bottom line in one or more of the following areas:[lxxxi]

- Productivity
- Development and/or sustainment cost
- Schedule
- Quality
- User Satisfaction
- Predictability

The Software Program Manager's Network is a Congressionally funded, tri-service technology transfer organization that is organized to provide direct support to DoD software-intensive programs. They have identified the "Best Practices" that are the

most frequently and successfully used in industry for large software-intensive system development and sustainment. These "Best Practices" were deemed essential for DoD development programs, were published in the April 1995 issue of *Netfocus* [lxxxii] and include the following 9 categories and descriptions:

1. FORMAL RISK MANAGEMENT

The discipline of risk management is vital to the success of any software effort. A formal risk management process requires corporate acceptance of risk as a major consideration for software program management, commitment of program resources, and formal methods for identifying, monitoring, and managing risk.

2. AGREEMENT ON INTERFACES

To deal with the chronic problem of vague, inaccurate and untestable specifications, the Council proposed that a baseline use interface must be agreed upon before the beginning of implementation activities, and that such user interface must be made and maintained as an integral part of the system specification. For those projects developing both hardware and software, a separate software specification must be written with an explicit and complete interface description.

3. FORMAL INSPECTIONS

Inspections should be conducted on requirements, architecture, designs at all levels(particularly detailed design), on code prior to unit test, and on test plans.

4. METRIC-BASED SCHEDULING AND MANAGEMENT

Statistical quality control and schedules should be maintained. This requires early calculation of size metrics, projection of costs and schedules from empirical patterns, and tracking of project status through the use of captured result metrics. Use of a parametric analyzer or other automated projection tool is also recommended.

5. BINARY QUALITY GATES AT THE INCH-PEBBLE LEVEL

Completion of each task in the lowest-level activity network needs to be defined by an objective binary indication. These completion events should be in the form of gates that assess either the quality of the products produced, or the adequacy and completeness of the finished process. Gates may take the form of technical reviews, completion of a specific set of tests which integrate or qualify software components, demonstrations, or project audits. The binary indication is meeting a predefined performance standard (e.g., defect density of less than four per function point). Activities are closed only upon satisfying the standard, with no partial credit given. Quality gates can be applied at any time during the project -- including solicitation.

6. PROGRAM-WIDE VISIBILITY OF PROGRESS VS. PLAN
The core indicators of project health or dysfunction should be made readily available to all project participants. Anonymous channel feedback should be encouraged to enable unfavorable news to move freely up and down project hierarchy.

7. DEFECT TRACKING AGAINST QUALITY TARGETS
Defects should be tracked formally at each project phase or activity. Configuration management (CM) enables each defect to be recorded and traced through to removal. In this approach there is no such thing as a private defect, that is, one detected and removed without being recorded. Initial quality targets (expressed, for example, in defects per function point) as well as to counts defects removed in order to track progress during testing activities.

8. CONFIGURATION MANAGEMENT
The discipline of CM is vital to the success of any software effort. CM is an integrated process for identifying, documenting, monitoring, evaluating, controlling, and approving all changes made during the life-cycle of the program for information that is shared by more than one individual or organization.

9. PEOPLE-AWARE MANAGEMENT ACCOUNTABLITY
Management must be accountable for staffing qualified people (those with domain knowledge and similar experience in previously successful projects) as well as for fostering an environment conducive to high morale and low voluntary staff turnover.

To support "Best Practices" the Support Program Manager's Network has developed a support organization that operates under the auspices of the following mission statement:

> *The Mission of the Software Program Managers Network is to enable managers of large-scale, software-intensive development or maintenance projects to more effectively manage and succeed by identifying and conveying to them management Best Practices, lessons-learned, and direct support.* [lxxxiii]

The "Best Practices" Initiative, jointly authored by Noel Longuemare, the Under Secretary of Defense for Acquisition & Technology, and Emmett Paige Jr., the Department of Defense Chief Information Officer, was based on the fact that that many effective practices exist for managing software – both in industry and government.

However, their use and understanding are not widespread within DoD software acquisition programs.

> *Current DoD software acquisition practices have not proven to provide an effective framework for managing the large-scale software development and maintenance that are an essential part of our increasingly complex weapons systems. Although many excellent practices for the effectively managing such programs exist in both industry and government, their understanding and use within our software acquisition programs is not widespread . . . We share these concerns, and together have established the* ***Software Acquisition Best Practices Initiative*** *to improve and restructure our software acquisition process.*
>
> *The purposes of this initiative are to:*
>
> - *Focus the Defense acquisition community on employing effective, high-leverage software management practices;*
> - *Enable Program Managers to focus their software management efforts on producing quality software, rather than on activities directed towards satisfying regulations that have grown excessively complex over time;*
> - *Enable Program Managers to exercise flexibility in implementing* ***Best Practices*** *within disparate corporate and program cultures; and,*
> - *Provide Program Managers and staff with the training and tools necessary to effectively use and achieve the benefits of these practices.*[lxxxiv]

IX. Software Improvement Process

A 1987 Defense Science Board report stated that: "Today's major problems with military software development are not technical problems, but management problems."[lxxxv] Moreover, The National Software Alliance has established that: "There is one element that can be attributed to every software success or failure without exception . . . Building software is a ***people thing!***" [lxxxvi] The success or failure in software depends on the skills, experience, and integrity of the people who build, implement, and maintain it."[lxxxvii] Therefore, it is important that the Department of

Defense "goes to school on" the comparative industry experiences and evaluates both the successful and unsuccessful commercial ventures.

The software improvement process is considered to be a challenging undertaking for any organization under even the best of circumstances. As a guide to organizations in assessing the strengths and weakness of their software development processes, the Carnegie Mellon University Software Engineering Institute developed the Capability Maturity Model (CMM[SM]). [lxxxviii] Additionally, Function Point Analysis can be directly mapped to CMM to describe and document functional requirements, quantify the size of a project and estimate its scope. David Lipton, in his article "Function Points and the SEI Capability Maturity Model", [lxxxix] depicts the process in detail. "Best Practices" methodologies also dovetail directly into the CMM model. The CMM is organized into five levels that characterize the organization's software process maturity.

- *1. **Initial.** The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.*
- *2. **Repeatable.** Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.*
- *3. **Defined.** The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.*
- *4. **Managed.** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.*
- *5. **Optimizing.** Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.*

Norm Brown of the Software Program Manager's Network reports that the Software Engineering Institute's analysis of the 379 organizations at 99 companies that

are advanced enough to have process improvement programs in place and that have conducted maturity assessments, 73 percent do not rate higher than Level One.[xc] The Department of Defense's **goal is to achieve maturity Level Three** (Defined Process) for in-house Central Design Activities/Software Design Activities and weapons system Software Support Activities.[xci] Current government acquisition practice **encourages** (but does not require) all bidders on software contracts to have a mature, well defined, standardized process.[xcii]

X. Technical Based Planning

To further exacerbate the management problem, Paul Strassmann of Strassmann Consulting, Inc., has stated that: " . . . management has largely abdicated the technology acquisition decision to experts and consultants. They, of course, have a vested interest to jump into buying more and better and sooner. Consequently, the general tendency is to buy, and when it doesn't work you just scrap it and go buy the next round."[xciii] This process of abdication has manifested itself as a myopic technology based planning methodology that has become prevalent throughout DoD. Unfortunately, in this era of the most effective use of available funds "just scrap it and go buy the next round" has ceased to be an option.

Technology based planning methodologies in the military have traditionally revolved around the Information Resource Management (IRM) personnel's perceived needs of the organization. These perceived needs are eventually translated into organization-wide requirements. The idea of "give it to Mikey, he'll eat it"[xciv] is senior management's perspective on the technical acquisition requirements process. And because these requirements are based purely on technology perspective, as Paul

Strassmann has stressed, they tend to be extremely limited in their scope. The technical personnel are, as general rule, educated in the technical computer arts and/or sciences; however, they are not generally not well schooled in the business or technical processes associated with the overall project. Technology for technology's sake is introduced as the solution to every problem. Effectively, would you really like your plumber to design and build your house? [xcv]

XI. Commercial Successes

The Motorola Iridium project is one of the definitive examples demonstrating how the use of "Best Practices" has a positive effect on development. Motorola's telecommunications network of 77 satellites, as originally planned, was named after the chemical element Iridium whose atomic number is 77. Currently, the $3.4 billion project has reduced the number of satellites to 66, arranged in six polar orbital planes. It is designed to permit call switching via satellite that bypasses the ground infrastructure. Additionally, a network of 11 ground stations around the world serves as the interconnect points between the Iridium satellites and the terrestrial networks. These interconnect points contribute to the complexity of the system. To route traffic properly, each satellite carries a set of stored routing tables from which new routing instructions are chosen every few minutes. Signals from the ground are transmitted in bursts, or "packets," each of which includes the address of the intended destination and the communications is completed.

The Iridium project rivals the complexity of the requirements' development and integration of most of the larger military software developments. The following statistics show the effect of using "Best Practices" on the Motorola project: [xcvi]

- Defect reduction per thousand lines of source code, from 3.5 to 1.
- Development time, from 10:21:00 to 0:44:41 (days:hours:minutes)
- Test time per thousand lines of source code, from 12.5 to 2.5 hours.

Additionally, Raytheon's Electronics Systems Division has reported the following improvements through the use of "Best Practices" to the Software Program Manager's Network: [xcvii]

- Productivity increased 180%.
- Rework percentage decreased from 44% to less than 10%
- Cost/Schedule predictability from 180% to within +/- 3%

XII. Conclusions and Recommendations

In conclusion, the state of affairs with the government and their relationship to software development organizations continues to be reminiscent of the horse trader who ends up with only the halter and lead rope. The Department of Defense is presently in the position of horse trading with someone who knows more about horses than they do. In fact, the government is considered the least desirable place of employment for software professionals. Until the government can define their software developments requirement and educate a core group of professionals that can accurately estimate the scope of the software to integrate those requirements, they will always end up without a horse after the negotiations are completed. Additionally, we have scrutinized the present state of both commercial and military software development and suggested possible solutions to the inherent management problems. The following solutions and recommendations need to be time phased with educational portion being implemented

first to develop the government's core group of professionals. These solutions and recommendations include:

1. The application of successful commercial software development management techniques, specifically "Best Practices" to military projects. Additionally, every effort must be made to evaluate the Department of Defense's internal software development organizations using the CMM as a means to improve the software development process.

2. Contractor incentives must be applied to ensure that "Best Practices" techniques are used for military projects. The incentives would include, but not be limited to, the independent verification by a third party vendor acting as a surrogate for the government of the application of mutually agreed upon "Best Practices" and the associated metrics as a condition of the letting the contract and issuing of bonuses.

3. Software Program Manager's Network must be used as a resource by program managers for implementation of "Best Practices" methodologies on their projects.

4. The Government must accurately estimate software size and thereby be able to identify and isolate rework and most importantly, don't pay for it.

5. There must be an extensive software management training education program for requirement analysis and software estimation for Department of Defense information and acquisition professionals. These educational requirements could be included and expanded from the existing Chief Information Officer certification program at the Information Resource Management College. The distribute of these trained professionals must be distributed throughout the program management community.

Appendix – Chief Information Officer Certificate Program

The CIO Certificate Program, sponsored by the DoD CIO, provides a source of graduate education for all federal CIO's to use in developing agency personnel. It is responsive to the requirements set forth in the Clinger-Cohen Act of 1996 and establishes an official certificate that serves as recognition that an individual has received education in the Federal CIO competencies.

CERTIFICATE REQUIREMENTS[xcviii]

Award of the certificate requires completion of eight 5-day intensive courses OR the Advanced Management Program (AMP) supplemented by a lessor number of intensive courses OR a combination of eight intensive and elective courses [Industrial College of the Armed Forces (ICAF) and National War College (NWC) students only]. Primary courses in six subject areas must be completed. Two of the primary areas must be Policy and Performance and Results-Based Management. The remaining two courses can be selected from either the primary or enrichment offerings for any subject area. Refer to the CIO Course listing for specific offerings.

Graduates of the AMP, a 14-week educational program which provides an integrated perspective of information management, receive credit for the following primary areas: Policy, Information Resources Strategic Planning, Process Improvement, and Acquisition. Additional credits may be earned for electives and the specialty track. The number of intensive courses required to complete the CIO certificate following AMP graduation depends on which electives and specialty track the student completed during the AMP.

ICAF and NWC students may complete part of the eight course requirement during their academic year by enrolling in selected electives. Remaining requirements may then be met by completing the necessary one-week intensive courses.

Regardless of approach the participant takes to complete the certificate, participants should confer with their supervisors to determine which subject areas and courses are most critical for their positions and organizations.

From time to time, the IRM College may replace/add/delete courses and/or subject areas. In cases where courses and/or subject areas are dropped, students will receive credit for courses they have already taken while in the program.

Participants will have up to four years from the date of acceptance to complete the program.

METHODOLOGY

The primary teaching methodology is the seminar format supplemented by guest speakers. Completion of student assessments is mandatory and may take various forms, from individual papers and projects to team projects and presentations. In some cases, requirements are completed after the formal instruction. In these cases, students have up to three weeks to complete the assignment.

The IRM College conducts all classes on the Ft. McNair campus. As deemed appropriate, some courses may be taught at remote sites or using distance learning format.

PROGRAM ELIGIBILITY

The program is open to federal civilians in the grades of GS/GM 13-15 and military officers in the grades of O5-O6. A bachelor's degree is required. At this time, applications are not accepted from industry or international students. Waivers may be requested for applicants who are no more than one grade lower than minimum requirements. Waivers may also be requested for the degree requirement.

KEY COMPETENCY SUBJECT AREA	AMP COURSES	INTENSIVE COURSES
1. Policy		
	PRIMARY	PRIMARY
	Core courses	NWC: New World of the CIO
	ENRICHMENT	ENRICHMENT
	Track: Critical Frameworks Underlying Public Policy (Previously Public Policy in the Information Age)	IWO: Information Operations
2. Information Resources Strategic Planning	PRIMARY	PRIMARY
	Core courses	IMP: Information Management Planning
3. Leadership / Management	PRIMARY	PRIMARY
	Elective: Innovative Thinking for the Information Age OR Elective: Third Wave Organizations	LDC: Leadership for the 21st Century OR HRI: Strategic Human Resources Issues for IT Organizations
	ENRICHMENT	
	Elective: Overload: the Paradox of	

	Information	
4. Process Improvement	PRIMARY Core courses ENRICHMENT Track: Best Practices in Change Management (Previously Best Practices in Process Improvement) Elective: System Dynamics: Dealing with Complexity Elective: Electronic Commerce	PRIMARY LTO: Reengineering Organizational Processes ENRICHMENT MAS: Evaluating Strategic Alternatives with Modeling and Simulation ECB: Electronic Commerce: Doing Business on the Information Highway
5. Capital Planning and Investment	PRIMARY Elective: IT Capital Planning (Previously Managing Information Technology Investments)	PRIMARY MTI: IT Capital Planning (Previously Managing Information Technology Investments)
6. Performance and Results-Based Management	PRIMARY Elective: Measuring Results of Organizational Performance ENRICHMENT Elective: Information Visualization	PRIMARY MOP: Measuring Results of Organizational Performance ENRICHMENT INV: Information Visualization
7. Technology Assessment	PRIMARY Core courses PLUS Track: Emerging Information Technologies ENRICHMENT Elective: The Information Highway Elective: Applying Multimedia Technology Elective: Computer Modeling & Simulation Elective: Virtual Reality for Managers Elective: Telecommunications Technologies	PRIMARY CST: Critical Information System Technologies (Previously Emerging Information Technologies) ENRICHMENT IHW: The Information Highway IDS: Improving Organizational Performance with Intelligent Decision Systems WEB: Strategic Management of Web Sites TEL: Telecommunications Technologies

8. Architectures	PRIMARY	PRIMARY
	Elective: Managing Information Architectures and Infrastructures	ARC: Managing Information Architectures and Infrastructures OR DMS: Data Management Strategies
9. Security	PRIMARY	PRIMARY
	Elective: Managing Information Security in a Networked Environment	SEC: Managing Information Security in a Networked Environment
		ENRICHMENT
		AII: Assuring the Information Infrastructure
		SAT: Management Information Security - Advanced Topics
10. Acquisition	PRIMARY	PRIMARY
	Core courses	ITA: IT Acquisition for the CIO
	ENRICHMENT	ENRICHMENT
	Track 4: Information Systems Acquisition	CAR 805: Contemporary Approaches to Acquisition Reform
	Elective: Future Directions in Software Management	IRM 303: Advanced Information Systems Acquisition
		SAM 301: Advanced Software Acquisition Management
		APMC: Information Technology Elective Track

---

[i] *Concept for Future Joint Operations – Expanding Joint Vision 2010,* OPR Joint Warfighting Center, Fort Monroe, VA, May 1997, pp. 35.

[ii] Brown, Norm, "Industrial-Strength Management Strategies", *IEEE Software,* Vol. 13, No. 4, July 1996, pp. 94.

[iii] Longuemare, Noel and Paige, Emmett, Jr., "MEMORANDUM FOR SECRETARIES OF THE MILITARY DEPARTMENTS, Subject: Software Acquisition Best Practices Initiative", dated July 8, 1994.

[iv] *Computer Dictionary,* Third Edition, Microsoft Press, Richmond, Washington, 1997.

[v] Jones, Rick, Director of Federal Strategies, Intel Corporation, Briefing to SSCF, University of Texas, Austin, March 16, 1999.

[vi] Barber, K. Suzanne, Briefing to The University of Texas College of Engineering, Engineering Foundation Advisory Council, February 20, 1990 at the University of Texas, Austin.
[vii] Brooks, Frederick P., Jr., "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer,* April 1987.
[viii] Brooks, Frederick P., Jr., "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer,* April 1987.
[ix] Jones, Capers, as quoted by Evelyn Richards, "Society's Demands Push Software to Upper Limits: More Computer Crisis Likely," *The Washington Post,* December 9, 1990.
[x] Parnas, David Lorge, "Software Aspects of Strategic Defense Systems," *American Scientist,* September-October 1985.
[xi] Gibbs, W. Wayt, "Software's Chronic Crisis," *Scientific American,* September 1994
[xii] *Contracting for Computer Software Development – Serious Problems Require Management Attention to Avoid Wasting Millions,* FGMSD-80-4, General Accounting Office, November 9, 1979.
[xiii] *Software Workers for the New Millennium: Global Competitiveness Hangs in the Balance,* National Software Alliance, Arlington, VA, January 1998, pp. 2-1.
[xiv] Silvestri, George T., "Employment Outlook: 1996-206; Occupational Employment Projections to 2006," *Monthly Labor Review,* U.S. Department of Labor, November 1997.
[xv] *Software Workers for the New Millennium: Global Competitiveness Hangs in the Balance,* National Software Alliance, Arlington, VA, January 1998, pp. 2-1.
[xvi] Student's are not choosing high-tech education, study says, http://cnn.com/TECH/computing/9904/26/cybered.01.ap, April 26, 1999.
[xvii] *Software Workers for the New Millennium: Global Competitiveness Hangs in the Balance,* National Software Alliance, Arlington, VA, January 1998, pp. 2-2.
[xviii] Ibid.
[xix] Beir, Jeffery R., "Managing Creatives: Our Creative Workers Will Excel – If we Let Them," speech presented at *Industry Week's* annual Managing for Innovation Conference, Chicago, Illinois, March 13, 1995.
[xx] Parnas, David Lorge, University of Texas Distinguished Lecture Series in Software Development and Software Engineering, "Software Engineering: The Unconsummated Marriage," December 8, 1998
[xxi] Ibid.
[xxii] Ibid.
[xxiii] Ibid.
[xxiv] Ibid.
[xxv] Ibid.
[xxvi] Ibid.
[xxvii] Glass, Robert L., *Software Conflict: Essays on the Art and Science of Software Engineering,* Yourdon Press, Englewood Cliffs, New Jersey, 1991.
[xxviii] Parnas, David Lorge, "Software Aspects of Strategic Defense Systems," *American Scientist,* September-October 1985.
[xxix] Brooks, Frederick P., Jr., "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer,* April 1987.
[xxx] Johnson, J., "Chaos: The dollar drain of IT project failures," *Application Development Trends,* January 1995, pp. 41-47.
[xxxi] Ibid.
[xxxii] http://www.standishgroup.com
[xxxiii] The Standish Group, "Chaos," ," http://www.standishgroup.com/chaos.html
[xxxiv] Ibid.
[xxxv] Ibid.
[xxxvi] Ibid.
[xxxvii] Ibid.
[xxxviii] Ibid.
[xxxix] Gibbs, W. Wayt, "Software's Chronic Crisis," *Scientific American,* September 1994.
[xl] Ibid.
[xli] Ibid.
[xlii] Ibid.

[xliii] *New Denver Airport Impact of the Delayed Baggage System,* GAO/RCED-95-35BR, General Accounting Office, October 1994.
[xliv] *Air Traffic Control: Status of FAA's Modernization Program,* GAO/RCED-95-175FS, General Accounting Office, May 1995.
[xlv] McGraw, Karen and Harbison, Karan, *User-Centered Requirements: The Scenario Based Engineering Process,* Lawrence Erlbaum Associates, Publishers, Mahwah, New Jersey, 1997.
[xlvi] Ibid.
[xlvii] Barber, K. Suzanne, Briefing to The University of Texas College of Engineering, Engineering Foundation Advisory Council, February 20, 1990 at the University of Texas, Austin.
[xlviii] Brooks, Frederick P., Jr., "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer,* April 1987.
[xlix] As reported in *Guidelines for Successful Acquisition and Management of Software Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems,* Department of the Air Force Software Technology Support Center, June 1996, Volume 1, pp. 1-58.
[l] DiNitto, Samuel, A., Jr., "Rome Laboratory," *Crosstalk,* Software Technology Support Center, June/July 1992.
[li] Boehm, Barry, *IEEE Software,* September 1987.
[lii] As reported in *Guidelines for Successful Acquisition and Management of Software Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems,* Department of the Air Force Software Technology Support Center, June 1996, Volume 1
[liii] Davis, Christine, "Best Practices Overview," as reported to the Software Program Managers Network, 4 December 1996
[liv] Solomon, Paul, "EV-Reducing Risk and Rework," as reported to the Software Program Managers Network, 17 September 1998.
[lv] Boehm, Barry, *IEEE Software,* September 1987.
[lvi] Brown, Norm, Briefing "Achieving Software Savings", Software Program Managers Network, 17 December 1998.
[lvii] Ibid.
[lviii] Defense Science Board, *Task Force on Military Software, Executive Summary,* September 1987.
[lix] McGrath, Frank, Briefing "16 Best Practices: Management and Technical Practices with High ROI in the Development and Sustainment of Large-scale Software-intensive Systems," *Software Programmer's Managers Network,* http://www.spmn.com
[lx] *Guidelines for Successful Acquisition and Management of Software Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems,* Department of the Air Force Software Technology Support Center, June 1996, Volume 1, pp. 1-20.
[lxi] "About Function Point Analysis," http://www.banniater.com/ifpug/home/docs/abfpa.html.
[lxii] Brown, Norm, "Industrial-Strength Management Strategies", *IEEE Software,* Vol. 13, No. 4, July 1996, pp. 95
[lxiii] Poster observed by the author in a software development organization in Washington, D.C.
[lxiv] Cetron, Marvin and Davies, Owen, *Probable Tomorrows: How Science and Technology Will Transform Our Lives in the Next Twenty Years,* St. Martin's Press, New York, 1997.
[lxv] Brooks, Frederick P., Jr., "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer,* April 1987.
[lxvi] McGrath, Frank, Briefing "16 Best Practices: Management and Technical Practices with High ROI in the Development and Sustainment of Large-scale Software-intensive Systems," *Software Programmer's Managers Network,* http://www.spmn.com
[lxvii] Based on EIA Industry Survey and OSD Press Release, 13 July 1994.
[lxviii] Brown, Norm, "Industrial-Strength Management Strategies", *IEEE Software,* Vol. 13, No. 4, July 1996, pp. 94
[lxix] *Software Workers for the New Millennium: Global Competitiveness Hangs in the Balance,* National Software Alliance, Arlington, VA, January 1998.
[lxx] Proceedings of the "Improving Software Acquisition Management Workshop," Defense Systems Management College, 13 February 1997.
[lxxi] Brewin, Bob, "Army increases stake in battlefield digitization," *Federal Computer Week,* February 1, 1999.

[lxxii] "Profile of the Army: A Reference Handbook," Association of the United States Army, Institute of Land Warfare, February 1997, pp. 30.
[lxxiii] Brewin, Bob, "Army increases stake in battlefield digitization," *Federal Computer Week*, February 1, 1999.
[lxxiv] Rodgers, Michael W., Briefing to the Senior Service College Fellows – University of Texas, Austin on 9 March 1999.
[lxxv] Cohen, William S., *Annual Report to the President and the Congress*, Government Printing Office, Washington, D.C., April 1997.
[lxxvi] Jones, Capers, *Patterns of Software System Failure and Success,* International Thomson Computer Press, Boston, MA, 1996, pp. 70.
[lxxvii] Office of the Under Secretary of Defense for Acquisition & Technology, *Report of the Defense Science Board Task Force on Acquiring Defense Software Commercially,* June 1994.
[lxxviii] Ibid.
[lxxix] Ibid.
[lxxx] Comaford, Christine, "The End of Innocence," *PC Week*, December 25, 1995
[lxxxi] McGrath, Frank, Briefing "16 Best Practices: Management and Technical Practices with High ROI in the Development and Sustainment of Large-scale Software-intensive Systems," *Software Programmer's Managers Network,* http://www.spmn.com
[lxxxii] *NetFocus* is the Software Management Newsletter published and distributed by the Software Program Mangers Network.
[lxxxiii] Software Program Managers Network Website: http://www.spmn.com
[lxxxiv] Longuemare, Noel and Paige, Emmett, Jr., "MEMORANDUM FOR SECRETARIES OF THE MILITARY DEPARTMENTS, Subject: Software Acquisition Best Practices Initiative", dated July 8, 1994.
[lxxxv] Office of the Under Secretary of Defense for Acquisition, *Report of the Defense Science Board Task Force on Military Software,* September 1987.
[lxxxvi] *Software Workers for the New Millennium: Global Competitiveness Hangs in the Balance,* National Software Alliance, Arlington, VA, January 1998, pp. 1-12.
[lxxxvii] Ibid.
[lxxxviii] Ibid., pp. 7-11 – 7-12.
[lxxxix] Lipton, David, "Function Points and the SEI Capability Maturity Model," http://www.qpmg.com/seicmm2.htm.
[xc] Brown, Norm, "Industrial-Strength Management Strategies", *IEEE Software,* Vol. 13, No. 4, July 1996, pp. 95.
[xci] *Guidelines for Successful Acquisition and Management of Software Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems,* Department of the Air Force Software Technology Support Center, June 1996, Volume 1, pp. 7-13.
[xcii] Ibid.
[xciii] Strassmann, Paul A., "Do U.S. Firms Spend Too Much on Information Technology?, Interview with Norm Alster," *Investor's Business Daily,* April 3, 1997
[xciv] Barber, K. Suzanne, Briefing to The University of Texas College of Engineering, Engineering Foundation Advisory Council, February 20, 1990 at the University of Texas, Austin.
[xcv] Ibid.
[xcvi] Brown, Norm, Briefing "Achieving Software Savings", Software Program Managers Network, 17 December 1998.
[xcvii] Ibid.
[xcviii] http://www.ndu.edu/irmc/ Academic Programs / DoD-CIO

Bibliography

Air Traffic Control: Status of FAA's Modernization Program, GAO/RCED-95-175FS, General Accounting Office, May 1995.

Beir, Jeffery R., "Managing Creatives: Our Creative Workers Will Excel – If we Let Them," speech presented at *Industry Week's* annual Managing for Innovation Conference, Chicago, Illinois, March 13, 1995.

Brewin, Bob, "Army increases stake in battlefield digitization," *Federal Computer Week,* February 1, 1999.

Brown, Norm, "Industrial-Strength Management Strategies", *IEEE Software,* Vol. 13, No. 4, July 1996.

Brown, Norm, "Achieving Software Savings", Software Program Managers Network, 17 December 1998.

Brooks, Frederick P., Jr., "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer,* April 1987.

Cetron, Marvin and Davies, Owen, *Probable Tomorrows: How Science and Technology Will Transform Our Lives in the Next Twenty Years,* St. Martin's Press, New York, 1997.

Cohen, William S., *Annual Report to the President and the Congress*, Government Printing Office, Washington, D.C., April 1997.

Comaford, Christine, "The End of Innocence," *PC Week*, December 25, 1995.

Computer Dictionary, Third Edition, Microsoft Press, Richmond, Washington, 1997.

Concept for Future Joint Operations – Expanding Joint Vision 2010, OPR Joint Warfighting Center, Fort Monroe, VA, May 1997.

Contracting for Computer Software Development – Serious Problems Require Management Attention to Avoid Wasting Millions, FGMSD-80-4, General Accounting Office, November 9, 1979.

Defense Science Board, *Task Force on Military Software, Executive Summary,* September 1987.

DeMarco, Tom and Lister, Timothy, *Software State-of-the-Art: Selected Papers,* Dorset House Publishing Co. Inc, New York, New York, 1990.

DiNitto, Samuel, A., Jr., "Rome Laboratory," *Crosstalk,* Software Technology Support Center, June/July 1992.

Gibbs, W. Wayt, "Software's Chronic Crisis," *Scientific American,* September 1994.

Guidelines for Successful Acquisition and Management of Software Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems, Volume 1 & 2, Department of the Air Force Software Technology Support Center, June 1996

Glass, Robert L., *Software Conflict: Essays on the Art and Science of Software Engineering,* Yourdon Press, Englewood Cliffs, New Jersey, 1991.

Johnson, J., "Chaos: The dollar drain of IT project failures," *Application Development Trends,* January 1995.

Jones, Capers, *Applied Software Management, Assuring Productivity and Quality – Second Edition,* McGraw-Hill, New York, New York, 1997.

Jones, Capers, *Patterns of Software System Failure and Success,* International Thomson Computer Press, Boston, MA, 1996.

Kanter, Jerome, *Managing with Information, Fourth Edition,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1992.

Libicki, Martin C., *Standards – The Rough Road to the Common Byte,* Center for Advanced Concepts and Technology Institute for National Strategic Studies, National Defense University, May 1995.

Lipton, David, "Function Points and the SEI Capability Maturity Model," http://www.qpmg.com/seicmm2.htm.

Longuemare, Noel and Paige, Emmett, Jr., "MEMORANDUM FOR SECRETARIES OF THE MILITARY DEPARTMENTS, Subject: Software Acquisition Best Practices Initiative", dated July 8, 1994.

Martin, James, *The Great Transition,* American Management Association, New York, New York, 1995.

McGrath, Frank, Briefing "16 Best Practices: Management and Technical Practices with High ROI in the Development and Sustainment of Large-scale Software-intensive Systems," *Software Programmer's Managers Network,* http://www.spmn.com

McGraw, Karen and Harbison, Karan, *User-Centered Requirements: The Scenario Based Engineering Process,* Lawrence Erlbaum Associates, Publishers, Mahwah, New Jersey, 1997.

New Denver Airport Impact of the Delayed Baggage System, GAO/RCED-95-35BR, General Accounting Office, October 1994.

Office of the Under Secretary of Defense for Acquisition & Technology, *Report of the Defense Science Board Task Force on Acquiring Defense Software Commercially,* June 1994.

Office of the Under Secretary of Defense for Acquisition, *Report of the Defense Science Board Task Force on Military Software,* September 1987.

Parnas, David Lorge, University of Texas Distinguished Lecture Series in Software Development and Software Engineering, "Software Engineering: The Unconsummated Marriage," December 8, 1998.

Parnas, David Lorge, "Software Aspects of Strategic Defense Systems," *American Scientist,* September-October 1985.

Proceedings of the "Improving Software Acquisition Management Workshop," Defense Systems Management College, 13 February 1997.

"Profile of the Army: A Reference Handbook," Association of the United States Army, Institute of Land Warfare, February 1997.

Reifer, Donald J., *Software Management, Fifth Edition,* IEEE Computer Society Press, Los Alamitos, California, 1997.

Software Workers for the New Millennium: Global Competitiveness Hangs in the Balance, National Software Alliance, Arlington, VA, January 1998.

Solomon, Paul, "EV-Reducing Risk and Rework," as reported to the Software Program Managers Network, 17 September 1998.

Strassmann, Paul A., "Do U.S. Firms Spend Too Much on Information Technology?, Interview with Norm Alster," *Investor's Business Daily,* April 3, 1997.

Silvestri, George T., "Employment Outlook: 1996-206; Occupational Employment Projections to 2006," *Monthly Labor Review,* U.S. Department of Labor, November 1997.

Turban, Efraim and Mclean, Ephraim and Wetherbe, James, *Information Technology For Management,* John Wiley & Sons, Inc., New York, New York, 1996.

Yourdon, Edward, *Death March – The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects,* Prentice Hall PTR, Upper Saddle River, New Jersey, 1997.

Yourdon, Edward, *Decline & Fall of the American Programmer,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1992.